ÖVNING 4

ÖVNING 1 AV 4 – GRUNDLÄGGANDE RÖRELSE MED PILTANGENTERNA

På den här övningen ska vi träna på att förflytta en figur med hjälp av piltangenterna.

1. Öppna filerna du förberedde förra veckan (characterControl.fla och Main_Character.as).
2. I Main_Character.as, klistra in följande kod:

```
package
{
	import flash.display.MovieClip;
	import flash.events.KeyboardEvent;
	import flash.ui.Keyboard;

	public class Main_Character extends MovieClip {

		//----------------------
		//Constructor
		public function Main_Character() {
			init();
		}

		//----------------------
		//Initiates application
		private function init():void {
			//Listens for key press
			stage.addEventListener(KeyboardEvent.KEY_DOWN, onKeyDownPress);
		}

		//----------------------
		//Handles when user presses keyboard button
		private function onKeyDownPress(e:KeyboardEvent):void {

			//Checks what button has been pressed
			if (e.keyCode == Keyboard.LEFT) {
				player.x -= 10; //Moves player to the left
			} else if (e.keyCode == Keyboard.RIGHT) {
				player.x += 10; //Moves player to the right
			} else if (e.keyCode == Keyboard.UP) {
				player.y -= 10; //Moves player up
			} else if (e.keyCode == Keyboard.DOWN) {
				player.y += 10; //Moves player down
			}
		}
	}
}
```

3. Spara och testkör applikationen. Om allt stämmer ska det nu gå att flytta din figur med piltangenterna.
4. Vad är det vi gjort? Det mesta känner du igen från tidigare övningar.
    a. Vi importerar de nödvändiga paketen.
    b. Konstruktorn kallar init()-metoden, vilken lägger till en event listener på scenen för att lyssna efter knapptryckningar.
    c. Event handlern kollar sedan med en if-sats vilka knappar som blivit intryckta och beroende på vilken knapp det är så flyttas spelaren en liten bit.

ÖVNING 2 AV 4 – GÖR OM, GÖR RÄTT

Det sättet du precis flyttat din figur på är inte bästa sättet att göra det, men syftade till att du skulle få förståelse för hur det fungerar. Nu ska vi göra om samma sak, fast på ”rätt” sätt.

1. Skapa en ny .AS-fil (döp den till Main_Character_Two.as). Byt Class i Properties på din characterControl.fla till Main_Character_Two.
2. Klistra in följande kod:

```
package
{
	import flash.display.MovieClip;
	import flash.events.KeyboardEvent;
	import flash.ui.Keyboard;
	import flash.events.Event;

	public class Main_Character_Two extends MovieClip {
		//Declare variables
		var vx:int;
		var vy:int;
		var velocity:uint;

		//--------------------
		//Constructor
		public function Main_Character_Two() {
			init();
		}

		//--------------------
		//Inilializes game
		function init():void {
			//Initialize variables
			vx = 0;
			vy = 0;
			velocity = 5;

			//Add event listeners
			stage.addEventListener(KeyboardEvent.KEY_DOWN, onKeyDownPress);
			stage.addEventListener(KeyboardEvent.KEY_UP, onKeyUpPress);
			stage.addEventListener(Event.ENTER_FRAME, onEnterFrameEvent);
		}

		//--------------------
		//Handles key down press
		function onKeyDownPress(e:KeyboardEvent):void {
			//Checks which button was pressed
			if (e.keyCode == Keyboard.LEFT) {
				vx = -velocity; //Moves player left
			} else if (e.keyCode == Keyboard.RIGHT) {
				vx = velocity; //Moves player right
			} else if (e.keyCode == Keyboard.UP) {
				vy = -velocity; //Moves player up
			} else if (e.keyCode == Keyboard.DOWN) {
				vy = velocity; //Moves player down
			}
		}

		//--------------------
		//Handles key release
		function onKeyUpPress(e:KeyboardEvent):void {
			//Resets variables values depending on key pressed (to stop player)
			if (e.keyCode == Keyboard.LEFT || e.keyCode == Keyboard.RIGHT) {
				vx = 0;
			} else if (e.keyCode == Keyboard.DOWN || e.keyCode == Keyboard.UP) {
				vy = 0;
			}
		}

		//--------------------
		//Handles enter frame
		function onEnterFrameEvent(e:Event):void {
			//Move the player (i.e. updates player position)
			player.x += vx;
			player.y += vy;
		}
	}
}
```

3. Spara och testa din applikation. Nu fungerar det mycket smidigare att styra din figur, och du kan dessutom förflytta den diagonalt. Det vi gjorde var:
    a. Vi importerade nödvändiga paket:

```
import flash.display.MovieClip;
import flash.events.KeyboardEvent;
import flash.ui.Keyboard;
import flash.events.Event;
```

    b. Vi deklarerade variabler:

```
var vx:int;
var vy:int;
var velocity:uint;
```

    Bokstaven v i namnen vx och vy står för velocity (hastighet). vx innebär med andra ord hastigheten på x-axeln. Lite senare i koden anger vi att variabeln velocity är 5. Detta innebär att det är 5 antal pixlar som du vill att din figur ska flytta per frame. Har du t.ex. en frame rate på 30 fps innebär det att figuren flyttar med en hastighet på 150 pixlar i sekunden (5x30).

    Anledningen till att vi har en variabel som heter velocity är för att det ska vara enkelt att ändra hastigheten. Skulle vi t.ex. vilja sätta hastigheten till 10 istället behöver vi bara ändra variabelns värde på ett ställe.

    c. Sedan satte vi initialvärden på variablerna och lade in event listeners:

```
//--------------------
//Inilializes game
function init():void {
                    //Initialize variables
                    vx = 0;
                    vy = 0;
                    velocity = 5;

                    //Add event listeners
                    stage.addEventListener(KeyboardEvent.KEY_DOWN, onKeyDownPress);
                    stage.addEventListener(KeyboardEvent.KEY_UP, onKeyUpPress);
                    stage.addEventListener(Event.ENTER_FRAME, onEnterFrameEvent);
}
```

    d. Vi lade in en event handler som hanterar när användaren trycker ner en knapp:

```
//--------------------
//Handles key down press
function onKeyDownPress(e:KeyboardEvent):void {
		//Checks which button was pressed
		if (e.keyCode == Keyboard.LEFT) {
			vx = -velocity; //Moves player left
		} else if (e.keyCode == Keyboard.RIGHT) {
			vx = velocity; //Moves player right
		} else if (e.keyCode == Keyboard.UP) {
			vy = -velocity; //Moves player up
		} else if (e.keyCode == Keyboard.DOWN) {
			vy = velocity; //Moves player down
		}
}
```

Med en if-sats kollar vi här vilken knapp som tryckts ner och sätter hastigheten som figuren ska röra sig. Ska figuren röra sig åt vänster måste vi skriva - först, och detsamma gäller när figuren ska röra sig uppåt (kommer du ihåg hur det fungerade med x och y?).

Här skiljer det sig från övningen ovan, för själva förflyttningen av figuren sker alltså inte i denna metod. Här har vi bara angett hur fort, och åt vilket håll den ska flytta sig.

e. Vi lade sedan in en event handler som hanterar när användaren släpper knappen de tryckt ner:

```
//--------------------
//Handles key release
function onKeyUpPress(e:KeyboardEvent):void {
		//Resets variables values depending on key pressed (to stop player)
		if (e.keyCode == Keyboard.LEFT || e.keyCode == Keyboard.RIGHT) {
			vx = 0;
		} else if (e.keyCode == Keyboard.DOWN || e.keyCode == Keyboard.UP) {
			vy = 0;
		}
}
```

När användaren släpper knappen nollställs hastigheten som figuren rör sig, d.v.s. den stannar, helt enkelt.

f. Slutligen skapade vi en event handler som hanterar enter frame-eventet, ett väldigt behändigt event:

```
//--------------------
//Handles enter frame
function onEnterFrameEvent(e:Event):void {
		//Move the player (i.e. updates player position)
		player.x += vx;
		player.y += vy;
}
```

En enter frame event avfyras varje gång en ny frame spelas upp – med andra ord, om du har en frame rate på 30 fps så avfyras en enter frame event 30 gånger per sekund. Eftersom vi använder vår enter frame event handler för att flytta figuren betyder det att 30 gånger varje sekund ställs spelarens position om. Om vx är -5 kommer alltså

figuren att flyttas -5 i sidled 30 gånger varje sekund. Det är detta som gör att figuren förflyttas så smidigt.

4. Testa att lägga in en trace() i din onEnterFrameEvent() för att se hur ofta den skriver ut någonting.
5. Se till att du förstått koden vi använt. Om du inte riktigt hängde med på vad alla funktioner innebär kan du testa att kommentera bort dem och se vad skillnaden blir. Du kan också lägga in trace() för att t.ex. se hur ofta ett event avfyras. Testa dig fram!
6. Testa att skriva om if/else-satsen i onKeyDownPress med en switch istället. Om du behöver hjälp på traven finns min lösning på nästa sida.

```
//Checks which button was pressed
switch(e.keyCode) {
    case Keyboard.LEFT:
        vx = -velocity;
        break;

    case Keyboard.RIGHT:
        vx = velocity;
        break;

    case Keyboard.UP:
        vy = -velocity;
        break;

    case Keyboard.DOWN:
        vy = velocity;
        break;

    default:
        break;
}
```

ÖVNING 3 AV 4 – INSKRÄNKA RÖRELSE

Det finns tre sätt att hantera att figuren rör sig utanför scenen:

- Blocking movement (förhindra att figuren rör sig utanför scenen)
- Screen wrapping (få figuren att dyka upp på motsatt sida som den försvann ifrån)
- Scrolling (förflytta bakgrunden för att imitera rörelse över en större yta)

BLOCKING MOVEMENT

1. Lägg till följande kod i din onEnterFrameEvent():

```
//Stop player at stage edges
if (player.x > stage.stageWidth) {
        player.x = stage.stageWidth;
} else if (player.x  < 0) {
        player.x = 0 ;
}

if (player.y < 0) {
        player.y = 0 ;
} else if (player.y  > stage.stageHeight) {
        player.y = stage.stageHeight;
}
```

2. Spara och testa din applikation. Det är inte en ultimat lösning för din figur försvinner halvvägs utanför kanten. Men vi kollar ändå på vad det är vi gjort:
    a. I den första if-satsen jämför vi figurens x-position med scenens bredd, och om figurens position är större (d.v.s. den är längre åt höger) än scenen sätter vi positionen till scenens bredd.
    b. Om figurens x-position är mindre än 0 (d.v.s. utanför scenens vänstra sida) sätter vi den till 0.
    c. I den andra if-satsen kollar vi om figurens y-position är mindre än 0 (d.v.s. högre upp än scenen), och i så fall nollställer vi den.
    d. Vi jämför figurens y-position med scenens höjd, och om den är högre än scenen (d.v.s. under scenens kant), likställer vi figurens position med scenens höjd.
3. Vad är då problemet med vår kod? Om du kommer ihåg från förra övningen så satte vi manuellt figurens x- och y-värden till -25. Det är detta som gör att figuren hamnar utanför scenen, eftersom vi beräknar dess position beroende på dess x- och y-värden. För att lösa detta måste vi räkna om spelarens position i vår jämförelse. Använd dig av följande variabler och försök själv komma på en lösning:

```
var playerHalfWidth:uint = player.width / 2;
var playerHalfHeight:uint = player.height / 2;
```

Behöver du hjälp på traven finns min lösning på nästa sida.

```
60          //--------------------
61          //Handles enter frame
62          function onEnterFrameEvent(e:Event):void {
63              //Move the player (i.e. updates player position)
64              player.x += vx;
65              player.y += vy;
66
67              var playerHalfWidth:uint = player.width / 2;
68              var playerHalfHeight:uint = player.height / 2;
69
70              //Stop player at stage edges
71              if ((player.x + playerHalfWidth) > stage.stageWidth) {
72                  player.x = stage.stageWidth - playerHalfWidth;
73              } else if (player.x - playerHalfWidth  < 0) {
74                  player.x = 0 + playerHalfWidth ;
75              }
76
77              if (player.y - playerHalfHeight < 0) {
78                  player.y = 0 + playerHalfHeight;
79              } else if (player.y + playerHalfHeight  > stage.stageHeight) {
80                  player.y = stage.stageHeight - playerHalfHeight;
81              }
82          }
```

SCREEN WRAPPING

1. Byt ut if-satserna i din onEnterFrameEvent() mot följande (vill du inte ta bort koden du skrivit kan du bara kommentera bort den inom /* */):

```
//Screen wrapping
if (player.x - playerHalfWidth > stage.stageWidth) {
        player.x = 0 - playerHalfWidth;
} else if (player.x + playerHalfWidth < 0) {
        player.x = stage.stageWidth + playerHalfWidth;
}

if (player.y + playerHalfHeight < 0) {
        player.y = stage.stageHeight + playerHalfHeight;
} else if (player.y - playerHalfHeight > stage.stageHeight) {
        player.y = 0 - playerHalfHeight;
}
```

2. Spara och testa din kod. Nu dyker din figur upp på andra sidan istället för att stanna vid kanten.
3. Vad är det vi gjort? Det fungerar nästan på samma sätt som den tidigare koden, men istället för att placera figuren vid kanten när den kommer dit så flyttar vi den till motsatt sida. På det sättet skapar vi en illusion av att figuren går runt.

SCROLLING

1. För den här uppgiften måste du gå in i characterControl.fla och skapa ett nytt lager. Lägg lagret längst ner och döp det till background. Lägg in Movie Clipet Background som du skapade på förra övningen på scenen (på lagret background) och ge det instansnamnet background.

2. Byt ut koden till din onEnterFrameEvent() mot denna (om du inte vill ta bort den gamla koden kan du kommentera bort den inom /* */):

```
function onEnterFrameEvent(event:Event):void {
		//Initialise local variables
		var playerHalfWidth:uint = player.width / 2;
		var playerHalfHeight:uint = player.height / 2;
		var backgroundHalfWidth:uint = background.width / 2;
		var backgroundHalfHeight:uint = background.height / 2;

		//Move the background
		background.x += - vx;
		background.y += - vy;

		//Stop background at stage edges
		if (background.x + backgroundHalfWidth < stage.stageWidth) {
			background.x = stage.stageWidth - backgroundHalfWidth;
		} else if (background.x - backgroundHalfWidth > 0) {
			background.x = 0 + backgroundHalfWidth;
		}

		if (background.y - backgroundHalfHeight > 0) {
			background.y = 0 + backgroundHalfHeight;
		} else if (background.y + backgroundHalfHeight < stage.stageHeight) {
			background.y = stage.stageHeight - backgroundHalfHeight;
		}
}
```

3. Spara och testa applikationen. Nu rör sig bakgrunden istället för figuren.
4. Vad är det vi gjort? Egentligen har vi gjort samma sak som då vi gjorde blocking movement tidigare, men på bakgrunden istället. Jämför koden med den tidigare. Ser du likheterna och skillnaderna?
5. Som du kanske ser använder vi negativ velocitet för bakgrunden, vilket vi inte gjorde för figuren. Detta är för att bakgrunden ska röra sig i motsatt riktning jämfört med figuren.

```
background.x += - vx;
background.y += - vy;
```

6. Som det är nu är dock inte koden helt perfekt. Figuren rör sig inte och bakgrunden börjar röra på sig trots att figuren inte är vid kanten. Vad vi behöver är en s.k. inner boundary, d.v.s. en inre begränsning för hur figuren kan röra sig utan att bakgrunden rör sig. Nedan presenteras färdig kod för hur en inner boundary kan byggas. Du får gärna försöka själv innan du kollar på koden, men testa oavsett att lägga till den nya koden i din nuvarande kod:

```
public class Main_Character_Two_Scrolling extends MovieClip {
		//Declare variables
		var vx:int;
		var vy:int;
		var velocity:uint;
		var rightInnerBoundary:uint;
		var leftInnerBoundary:uint;
		var topInnerBoundary:uint;
		var bottomInnerBoundary:uint;

		//--------------------
		//Constructor
		public function Main_Character_Two_Scrolling() {
				init();
		}

		//--------------------
		//Inilializes game
		function init():void {
				//Initialize variables
				vx = 0;
				vy = 0;
				velocity = 8;
				rightInnerBoundary = (stage.stageWidth / 2) + (stage.stageWidth / 4);
				leftInnerBoundary = (stage.stageWidth / 2) - (stage.stageWidth / 4);
				topInnerBoundary = (stage.stageHeight / 2) + (stage.stageHeight / 4);
				bottomInnerBoundary = (stage.stageHeight / 2) - (stage.stageHeight / 4);

				//Add event listeners
				stage.addEventListener(KeyboardEvent.KEY_DOWN, onKeyDownPress);
				stage.addEventListener(KeyboardEvent.KEY_UP, onKeyUpPress);
				stage.addEventListener(Event.ENTER_FRAME, onEnterFrameEvent);
		}
```

```
		//--------------------
		//Handles enter frame
		function onEnterFrameEvent(event:Event):void {
				//Initialise local variables
				var playerHalfWidth:uint = player.width / 2;
				var playerHalfHeight:uint = player.height / 2;
				var backgroundHalfWidth:uint = background.width / 2;
				var backgroundHalfHeight:uint = background.height / 2;

				//Move the player
				player.x += vx;
				player.y += vy;

				//Stop player at inner boundary edges
				if (player.x - playerHalfWidth < leftInnerBoundary) {
						player.x = leftInnerBoundary + playerHalfWidth;
						rightInnerBoundary = (stage.stageWidth / 2) + (stage.stageWidth / 4);
						background.x -= vx;
				} else if (player.x + playerHalfWidth > rightInnerBoundary) {
						player.x = rightInnerBoundary - playerHalfWidth;
						leftInnerBoundary = (stage.stageWidth / 2) - (stage.stageWidth / 4);
						background.x -= vx;
				}

				if (player.y - playerHalfHeight < topInnerBoundary) {
						player.y = topInnerBoundary + playerHalfHeight;
						bottomInnerBoundary = (stage.stageHeight / 2) + (stage.stageHeight / 4);
						background.y -= vy;
				} else if (player.y + playerHalfHeight > bottomInnerBoundary) {
						player.y = bottomInnerBoundary - playerHalfHeight;
						topInnerBoundary = (stage.stageHeight / 2) - (stage.stageHeight / 4);
						background.y -= vy;
				}

				//Stop background at stage edges
				if (background.x + backgroundHalfWidth < stage.stageWidth) {
						background.x = stage.stageWidth - backgroundHalfWidth;
						rightInnerBoundary = stage.stageWidth;
				} else if (background.x - backgroundHalfWidth > 0) {
						background.x = backgroundHalfWidth;
						leftInnerBoundary = 0;
				}

				if (background.y - backgroundHalfHeight > 0) {
						background.y = backgroundHalfHeight;
						topInnerBoundary = 0;
				} else if (background.y + backgroundHalfHeight < stage.stageHeight) {
```

```
                                        background.y = stage.stageHeight - backgroundHalfHeight;
                                        bottomInnerBoundary = stage.stageHeight;
                                }
                        }
}
```

7. Spara och testa din applikation. Om allt är som det ska så bör bakgrunden bara börja röra sig när din figur är en bit på väg mot scenens kant, och när du når bakgrundens slut ska figuren gå hela vägen till scenens kant.
8. Det vi gjort är:
    a. Vi börjar med att deklarera variabler, en för varje gräns (boundary):

```
var rightInnerBoundary:uint;
var leftInnerBoundary:uint;
var topInnerBoundary:uint;
var bottomInnerBoundary:uint;
```

    b. Vi måste sedan räkna ut var varje gräns ska gå. Testa gärna att ändra dessa siffror och se vad skillnaden blir:

```
rightInnerBoundary = (stage.stageWidth / 2) + (stage.stageWidth / 4);
leftInnerBoundary = (stage.stageWidth / 2) - (stage.stageWidth / 4);
topInnerBoundary = (stage.stageHeight / 2) + (stage.stageHeight / 4);
bottomInnerBoundary = (stage.stageHeight / 2) - (stage.stageHeight / 4);
```

    c. Sedan lägger vi in koden för att flytta figuren, precis som förut:

```
//Move the player
player.x += vx;
player.y += vy;
```

    d. Nu kommer vi till det som kan verka mest komplicerat. Vi jämför figurens position med den gräns vi bestämt (istället för att jämföra med scenens storlek, vilket vi gjorde förut) och om t.ex. positionen är mindre än den vänstra gränsen (d.v.s. figuren rör sig utanför gränsen åt vänster), så sätter vi figurens position till gränsen plus halva figurens storlek. Detta är exakt samma som tidigare också, fast med leftInnerBoundary istället för scenens storlek.

    Något vi dock måste göra här är att återställa rightInnerBoundary, eftersom den i vissa fall ändras (då spelaren går ända ut till kanten när bakgrunden tar slut), något som kommer senare i koden.

    Vi flyttar även bakgrundens position (jämför gärna med den gamla koden).

```
//Stop player at inner boundary edges
if (player.x - playerHalfWidth < leftInnerBoundary) {
                player.x = leftInnerBoundary + playerHalfWidth;
                rightInnerBoundary = (stage.stageWidth / 2) + (stage.stageWidth / 4);
                background.x -= vx;
} else if (player.x + playerHalfWidth > rightInnerBoundary) {
                player.x = rightInnerBoundary - playerHalfWidth;
                leftInnerBoundary = (stage.stageWidth / 2) - (stage.stageWidth / 4);
                background.x -= vx;
}
```

    e. Vi fortsätter med att göra exakt samma sak som ovan, fast lodrätt istället för vågrätt (d.v.s. y-position):

```
if (player.y - playerHalfHeight < topInnerBoundary) {
                    player.y = topInnerBoundary + playerHalfHeight;
                    bottomInnerBoundary = (stage.stageHeight / 2) + (stage.stageHeight / 4);
                    background.y -= vy;
} else if (player.y + playerHalfHeight > bottomInnerBoundary) {
                    player.y = bottomInnerBoundary - playerHalfHeight;
                    topInnerBoundary = (stage.stageHeight / 2) - (stage.stageHeight / 4);
                    background.y -= vy;
}
```

f. Slutligen lägger vi till ”anpassningar” av våra gränsvärden i den kod som flyttar på bakgrunden. Detta är för att figuren ska kunna gå ända till scenens kant när vi nått bakgrundens slut. Det är dessa gränsvärden vi återställer i koden i d och e – ifall användaren skulle flytta tillbaka figuren och vi måste återinföra den osynliga gränsen.

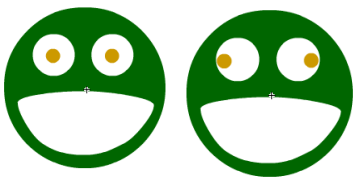
```
rightInnerBoundary = stage.stageWidth;

leftInnerBoundary = 0;

topInnerBoundary = 0;

bottomInnerBoundary = stage.stageHeight;
```

9. Se till att du förstår hur koden du precis använt fungerar – kanske behöver du läsa det flera gånger. Testa att ändra värdena för dina gränser och se vad skillnaden blir. Kommentera bort raderna där vi återställer gränsvärdena. Vad händer?

ÖVNING 4 AV 4 – FÖRBEREDELSE INFÖR NÄSTA ÖVNING

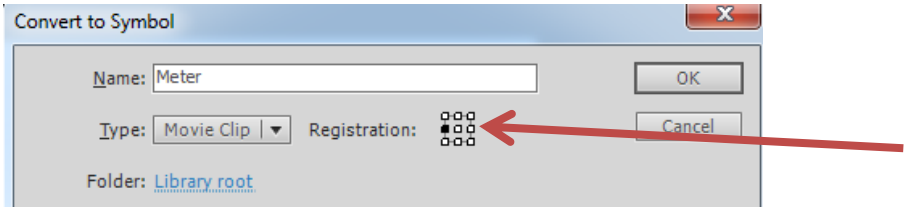
Nästa övning kommer även den att handla om att flytta på figurer med piltangenterna, dock behöver vi förbereda lite fler symboler denna gång.

1. Skapa en ny .FLA-fil som du döper till interactivePlayground. Skapa en ny .AS-fil som du döper till Main_Playground. I interactivePlayground, ange Main_Playground under Class i Properties.
2. Lägg en ruta som har bakgrundsfärg (blå) och rita en kulle (grön). Gör om kullen till ett Movie Clip, döp det till Hill och ge det instansnamnet hill.
3. Skapa två nya Movie Clip: Enemy och Player. Rita båda figurerna (50x50 pixlar stora som i förra övningen). Båda symbolerna ska ha två frames, och det ska ske någon slags förändring mellan de olika framesen. Till exempel:



4. Dra ut symbolerna på scenen och ge dem instansnamnen enemy respektive player.
5. Skapa ett nytt Movie Clip. Döp det till Health. Rita en avlång rektangel med både fyllnad och ram (ramen med en tjocklek på 3 px). Fyllnaden och ramen bör ha olika färg. Se till att Object drawing **inte** är markerat när du ritar.

6. Använd Selection tool (V) och markera din fyllnad (**inte ramen**). Välj Modify → Convert to symbol. Döp symbolen till Meter, typ Movie Clip. Sätt även Registration i mitten till vänster istället för default i övre vänstra hörnet. Ge ditt nya Movie Clip instansnamnet meter.



7. Du har nu gjort det möjligt att med kod kontrollera fyllnaden, och den kommer som du kanske redan gissat att användas som en ”hälsomätare”. Dra ut Health på scenen och ge den instansnamnet health.
8. Dra ut ett dynamiskt textfält på scenen (Classic Text), med rum för minst 2 rader. Ge det instansnamnet messageDisplay.
9. Dra ut ett mindre, dynamiskt textfält (Classic Text) och ge det instansnamnet scoreDisplay.
10. Skapa ett nytt Movie Clip, döp det till Apple. Rita ett äpple som är hälften så stort som dina figurer (25x25). Dra ut den på scenen och ge den instansnamnet apple.
11. Skapa en knapp, döp den till ReplayButton. Rita grafiken (skriv t.ex. ”Starta om spelet”). Dra dock inte ut knappen på scenen.
12. Skapa ett Movie Clip, döp det till Wall och rita en vägg (t.ex. en rektangel med ”tegelstenar”). Dra inte ut det på scenen.
13. Så här ser min lösning ut: